```
package finalproject.tests;

import static org.junit.Assert.assertEquals;
import org.junit.BeforeClass;
import org.junit.Test;
import org.junit.FixMethodOrder;
import org.junit.runners.MethodSorters;


import finalproject.Board;

@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class BoardTest{
    private static Board board;

    /**
     * Sets up reused objects for all test cases
     * @throws Exception
     */
    @BeforeClass
    public static void setupBeforeClass() throws Exception {
        board = new Board();
        board.initialize();
    }

    @Test
    public void testBoard() {
        assertEquals(board.getBoardLength(), 8);
        assertEquals(board.getBoardWidth(), 8);
    }

    /*
    @Test
    public void testInitialize() {
        // Check to see if hash maps initialized appropriately
    }


    @Test
    void testCheckKingCheckMate() {
        // Check to see when method written
    }
    */

    @Test
    public void testDisplay(){
        String expectedResult =
                  "|wR|wK|wB|wQ|wK|wB|wK|wR|\n"
                + "|wP|wP|wP|wP|wP|wP|wP|wP|\n"
                + "|  |  |  |  |  |  |  |  |\n"
                + "|  |  |  |  |  |  |  |  |\n"
                + "|  |  |  |  |  |  |  |  |\n"
                + "|  |  |  |  |  |  |  |  |\n"
                + "|bP|bP|bP|bP|bP|bP|bP|bP|\n"
                + "|bR|bK|bB|bK|bQ|bB|bK|bR|\n";
```

```
        assertEquals(expectedResult, board.display());
    }

    @Test
    public void testPlacePiece() {

        // test to see if basic invalid pawn move does not work
        board.placePiece(board.getPiece(1, 0), 1, 1);
        assertEquals(board.getPiece(1, 0).getType(), "Pawn");
        assertEquals(board.getPiece(1, 0).getColor(), "white");

        // test to see if basic valid pawn move does not work
        board.placePiece(board.getPiece(1,0),3,0);
            // Check there is a white pawn in the new location
        assertEquals(board.getPiece(3, 0).getType(), "Pawn");
        assertEquals(board.getPiece(3, 0).getColor(), "white");
            // Check that there is no longer a white pawn in the previous location
        assertEquals(board.isSpaceOccupied(1, 0), false);

        // need to develop a scenario to test occupied pieces
    }

}
```